COMP 508 Term Project

SYKario

Sanaz Taheri – Yahya Hassanzadeh

Problem definition

This project is about **sorting Karyotypes** in a way that the result is self-explanatory. **Karyotypes** means the number and visual appearance of the chromosomes in the cell nuclei of an organism or species. Most of the time (almost always) Karyotypes contain chromosomes positioned in a disordered manner i.e. they are rotated, they have overlap and there is no predefined and uniform distribution for them and etc. An example of Karyotype is given in the Figure1:

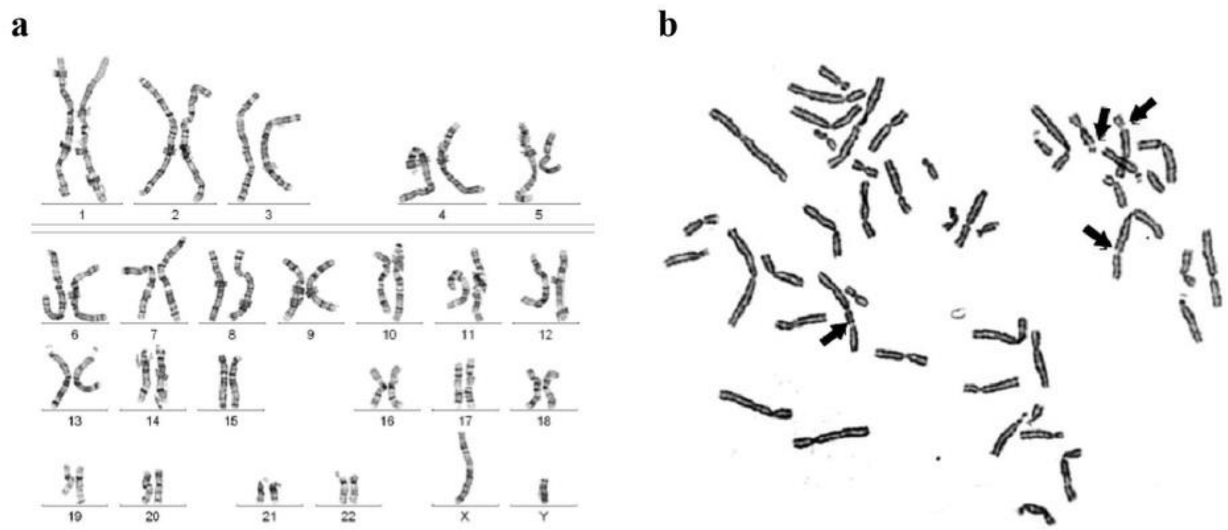
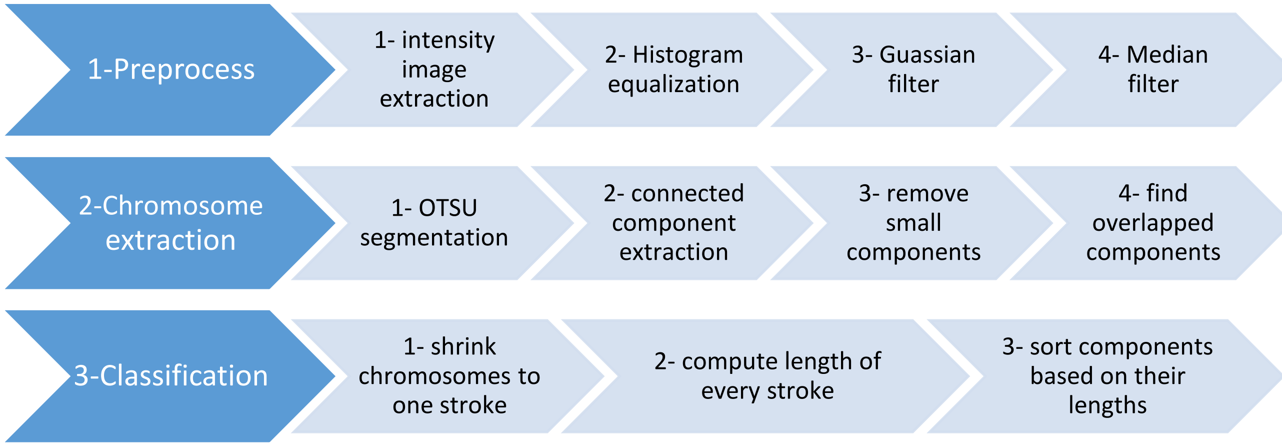


Figure 1) a)Sorted Chromosomes of Karyotype b)Karyotype

Our goal is to take the Karyotype like Figure1.b as the input and create another image having all the chromosomes in **a sorted shape** based on their **length**. Every chromosome should also be paired with its most matching one. The most matching one is the one with **strictly similar size**.

Solution

Our solution regarding to this problem can be summarized in the following flowchart:



1. Preprocess: In this step we try to increase the quality of image in order to extract more accurate features from it.

1.1. First, all the images should change to their gray levels. Matlab function for this purpose is **rgb2gray**.

1.2. To improve the distinguishability between chromosomes and the image background we apply the histogram equalization method. It also improves the inner connectivity of every chromosome. We used **adapthiseq** function of matlab built in functions.

1.3. To reduce noises, Gaussian filter has been utilized. We made a filter using **fspecial** and apply it on the image by the use of **medfilt2.**

1.4. some parts of the chromosomes after OTSU segmentation show up unintended holes inside of the chromosome, median filter has been applied to prevent it (It can recover the value of a pixel by the use of its neighborhood). Matlab's **medfilt2** does this filter for us.

Figure 3) KaryoType

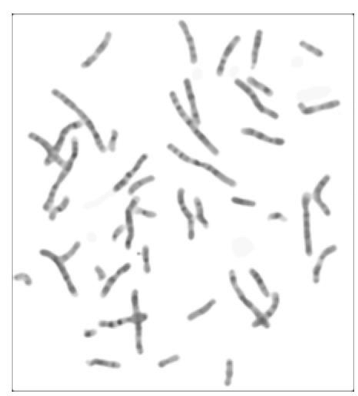
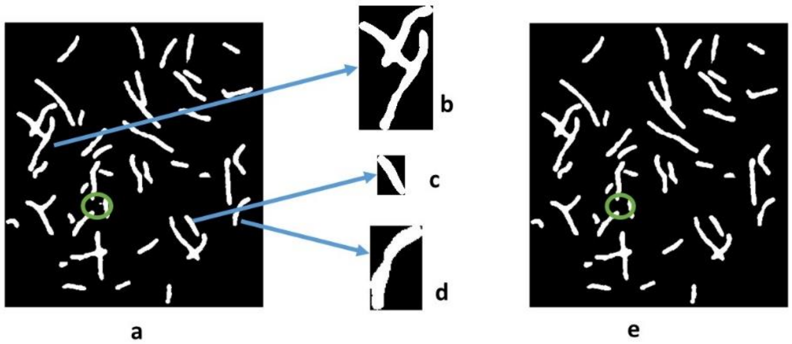


Figure 2) KaryoType after Preprocessing

2. Chromosome extraction:

2.1. We use OTSU segmentation to separate chromosomes from background. Indeed, since the intensity difference between chromosomes and background is noticeable, OTSU gives us a good result.

Graythresh of matlab can find the proper threshold and **im2bw** applies the segmentation based on that threshold



2.2. In the next step, we find every connected component which often corresponds to one chromosome. Sometimes, since there exists overlap between chromosomes, all of them are extracted as one component however is not correspond to one chromosome. We will make them separate in part 2.4. **regionprops** of Matlab has the functionality of finding and extraction of connected components.

2.3. Because of possible remaining errors resulting from the image noises, there may be some unrelated components which are extracted as chromosomes. They are all too small in comparison with the other components. We remove them before classification starts. We check whether the size of each component (total number of pixels) can satisfy a given threshold. Matlab's **Bwareaopen** function can remove the small connected components based on the given threshold as its input.

2.4. In order to find the overlapped chromosomes, we did a few searches, there were some proposed methods using edge based segmentation. We finally proposed and implemented a simple overlapped chromosome extractor using the skeleton which was easier to implement. Following is our proposed strategy to detect and separate chromosomes with overlap [1, 2]. We implemented this method as the Chromosome Separator (CE.m) function.

2.4.1. This CE function receives a connected component which consist of one or more chromosomes.

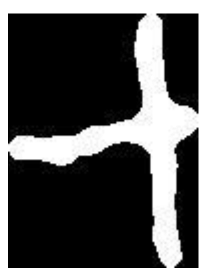


Figure 4- Input of CE function

2.4.2. CE first shrinks this component to a minimally connected strokes via bwmorph[3] function of MATLAB using its thin parameter. These connected strokes from a line for each chromosome. For example, image below corresponds to two overlapped chromosomes:

Figure 5-Converting the component to connected strokes. These strokes form one or more lines. Each line corresponds to a single chromosome.

2.4.3. It then extracts the skeleton of that connected component via bwmorph function of MATLAB using its skel parameter.

2.4.4. Next, CE finds endpoints of the connected lines and intersection point of them obtained from pervious steps bwmorph function of MATLAB using its endpoints and branchpoints parameters, respectively:

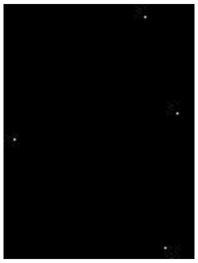


Figure 6-Endpoints of the lines

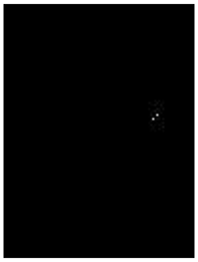


Figure 7- Branch points

2.4.5. Then CE calls the Line Tracker function (LT.m) by passing the connected strokes binary image and one of the endpoints. LT starts from that endpoint and follows the structure to track the line. When LT reaches other endpoint, terminates and extracts the line from input image. It returns the extracted line and also the input image without that line. CE then stores the line in its return array (C array) and recalls the LT with the second part –part with extracted line removed from- and does the same procedure until the second part is an empty binary image.

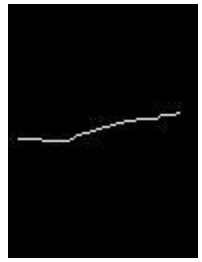


Figure 8- Extracted line by LT (The line return value)

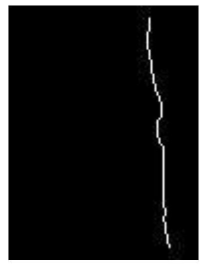


Figure 9- Input image with the extracted line removed from it(The rest return value)

As was described above CS (Chromosome Separator) is the heart of this overlap detection system. It receives a connected component and its skeleton and detects the single chromosomes of that connected component. Output of this function is the C array. Each element of this array represents a chromosome and has two fields: line and image. Line is the linear representation of that chromosome and image is the binary image of the chromosome:

Figure 10-Binary image of the first extracted chromosome

Figure 11-Binary image of second extracted chromosome

CS converts the connected strokes representation to binary image using the Chromosome Extractor (CE.m) function. This function receives the connected strokes representation and the connected components (Input of CS) and finds corresponding chromosome of the strokes representation.

Our line tracker strategy:

As was stated above, we developed a line tracker method. This method receives one endpoint of a line and extracts it from a binary image. It starts by the endpoint and checks the 8 neighborhood (Ad matrix) of each pixel to find the next pixel to go. It also keeps track of direction by updating a 3*3 matrix (Ac matrix). For each direction it moves, it increases the exact direction by 1 and 2-neighborhoods by 0.5. So whenever LT faces multiple choices to follow as its next move, it chooses the one with the highest value in Ac matrix. Followings show a sample update in Ac after moving in a certain direction.

		→

		+0.5
		+1
		+0.5

←		

+0.5		
+1	+0.5	

3. When the components are found, we should classify them to group of two (the most matching ones in the term of length).

3.1. We shrink chromosomes to a minimally connected stroke. Matlab's **Bwmorph** function with the '**thin'** parameter does the same thing.

3.2. The length of each stroke is a measure for the chromosome length. The length is extracted as the number of pixel in the stroke.

3.3. Finally, all the chromosomes will be sorted based on the stroke length. Every two consecutive chromosomes in the sorted list are assumed to be the most matching ones.

Results

As the result we sort the chromosomes of a given picture based on their sizes.

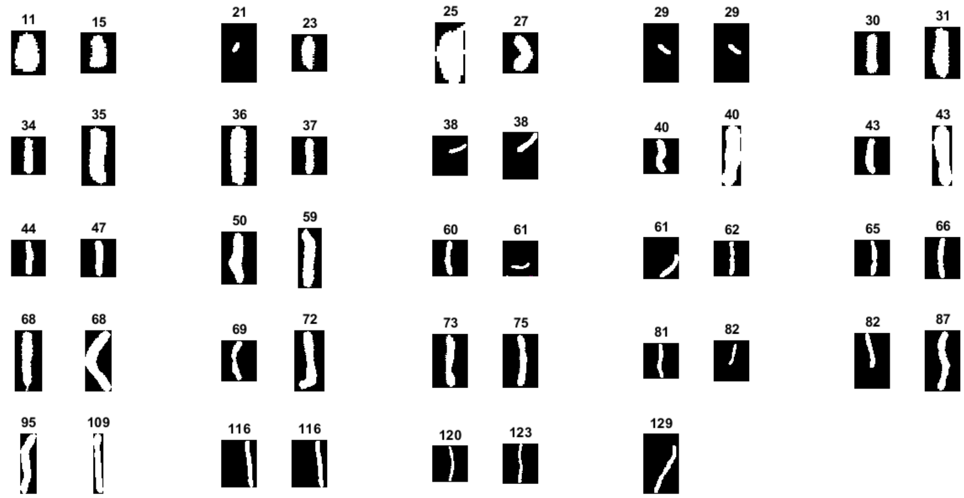


Figure 12)The sorted results of Figure3

We evaluate the result based on:

1. The error in the number of extracted chromosomes e.g. the difference between accurate number of chromosomes i.e. 46 and the number of extracted chromosomes
2. We calculate the length difference of every paired chromosomes. We get the average of this distance over all the paired chromosomes as one of the accuracy measurement.
3. The percentage of correct detection and extraction of overlapped chromosomes.

The following table (Figure 13) contains the result of our experiment over 10 samples of Karyotype. The first row and second row correspond to the result based on the first and second measurement respectively.

1-Number of extracted chromosomes error	-2	-2	-1	-4	0	-1	-1	-1	0	-1
2-Distance Average	3.0417	3.0417	2.5833	2.4000	5.2609	3.0833	2.8750	2.2500	5.2609	2.4000

Figure 14)Results

On the average we can detect and extract **2 overlapped** chromosomes with 57% of accuracy and **3 overlapped** chromosomes with **28%** of accuracy.

References:

1. Saiyod, S. and P. Wayalun, *A New Technique for Edge Detection of Chromosome G-BAND Images for Segmentation*, in *Advanced Approaches to Intelligent Information and Database Systems*. 2014, Springer. p. 315-323.
2. Ji, L., *Fully automatic chromosome segmentation.* Cytometry, 1994. **17**(3): p. 196-208.
3. mathworks. *Morphological operations on binary images*. 2015.